8086 Microprocessor

8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20address lines and16 data lines that provides up to 1MB storage. It consists of powerful instruction set, which provides operations like multiplication and division easily. It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

Features of 8086

The most prominent features of a 8086 microprocessor are as follows –

* It has an instruction queue, which is capable of storing six instruction bytes from the memory resulting in faster processing.
* It was the first 16-bit processor having 16-bit ALU, 16-bit registers, internal data bus, and 16-bit external data bus resulting in faster processing.
* It is available in 3 versions based on the frequency of operation −

1. 8086 → 5MHz
2. 8086-2 → 8MHz
3. (c)8086-1 → 10 MHz

* It uses two stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.
* Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue.
* Execute stage executes these instructions.
* It has 256 vectored interrupts.
* It consists of 29,000 transistors.

Architecture of 8086

Microprocessor is divided into two functional units, i.e., EU (Execution Unit) and BIU (Bus Interface Unit).

**EU (Execution Unit)** Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

It has the following functional parts –

**ALU** It handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.

**Flag Register** It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups Conditional Flags and Control Flags.

**Conditional Flags** It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags

* + - * 1. Carry flag − This flag indicates an overflow condition for arithmetic operations.
        2. Auxiliary flag − When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
        3. Parity flag − This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1’s, then the Parity Flag is set. For odd number of 1’s, the Parity Flag is reset.
        4. Zero flag − This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
        5. Sign flag − This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
        6. Overflow flag − This flag represents the result when the system capacity is exceeded.

**Control Flags** controls the operations of the execution unit. Following is the list of control flags

1. Trap flag − It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
2. Interrupt flag − It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
3. Direction flag − It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.

**General purpose register** There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively

* AX register − It is also known as accumulator register. It is used to store operands for arithmetic operations.
* BX register − It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
* CX register − It is referred to as counter. It is used in loop instruction to store the loop counter.
* DX register − This register is used to hold I/O port address for I/O instruction.

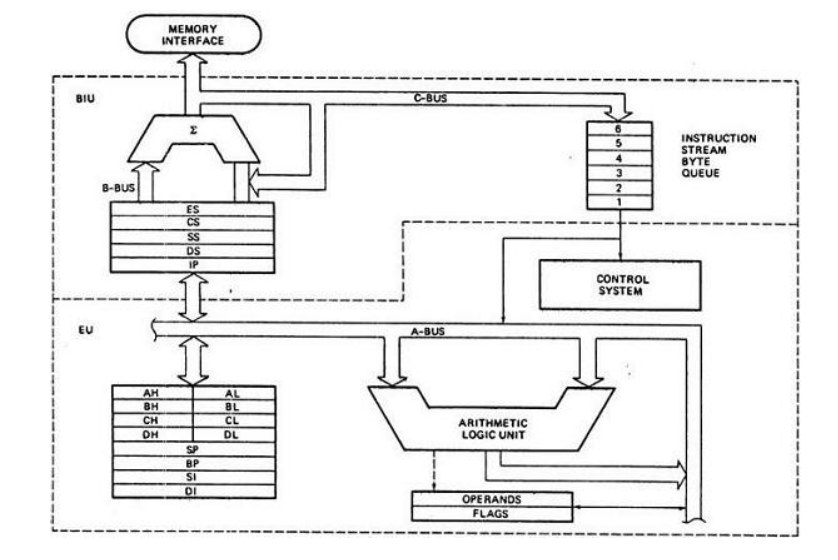
**Stack pointer register** It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

**BIU (Bus Interface Unit)** BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

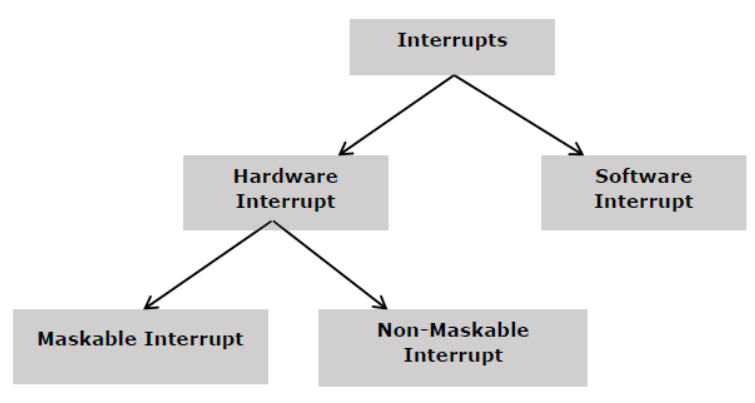
The Bus Interface Unit (BIU) generates the 20-bit physical memory address and provides the interface with external memory (ROM/RAM). 8086 has a single memory interface. To speed up the execution, 6- bytes of instruction are fetched in advance and kept in a 6-byte Instruction Queue while other instructions are being executed in the Execution Unit (EU). Hence after the execution of an instruction, the next instruction is directly fetched from the instruction queue without having to wait for the external memory to send the instruction. This is called pipe-lining and is helpful for speeding up the overall execution process.

It has the following functional parts –

* Instruction queue − BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.
* Fetching the next instruction while the current instruction executes is called pipelining.
* Segment register − BIU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.
  + CS − It stands for Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.
  + DS − It stands for Data Segment. It consists of data used by the program andis accessed in the data segment by an offset address or the content of other register that holds the offset address.
  + SS − It stands for Stack Segment. It handles memory to store data and addresses during execution.
  + ES − It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.
* Instruction pointer − It is a 16-bit register used to hold the address of the next instruction to be executed.



The following image shows the types of interrupts we have in a 8086 microprocessor



**Hardware Interrupts**

Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor. The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

**NMI**

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place –

• Completes the current instruction that is in progress.

• Pushes the Flag register values on to the stack.

• Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.

• IP is loaded from the contents of the word location 00008H.

• CS is loaded from the contents of the next word location 0000AH.

• Interrupt flag and trap flag are reset to 0.

**INTR**

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.

The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends ‘0’ on INTA pin twice. The first ‘0’ means INTA informs the external device to get ready and during the second ‘0’ the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor –

* First completes the current instruction.
* Activates INTA output and receives the interrupt type, say X.
* Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
* IP value is loaded from the contents of word location X × 4
* CS is loaded from the contents of the next word location.
* Interrupt flag and trap flag is reset to 0

**Software Interrupts**

Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes “INT- Interrupt instruction with type number” It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.

**Addressing modes**

There are 8 different addressing modes

* **Immediate addressing mode** The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.
* **Register addressing mode** It means that the register is the source of an operand for an instruction.
* **Direct addressing mode** The addressing mode in which the effective address of the memory location is written directly in the instruction.
* **Register indirect addressing mode** This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI.
* **Based addressing mode In this addressing mode**, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement.
* **Indexed addressing mode** In this addressing mode, the operands offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements.
* **Based-index addressing mode** In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.
* **Based indexed with displacement mode** In this addressing mode, the operands offset is computed by adding the base register contents. An Index registers contents and 8 or 16-bit displacement.